

Multivariate Polynomial Integration and Derivative Are Polynomial Time Inapproximable Unless $P=NP^*$

Bin Fu

Department of Computer Science
University of Texas-Pan American
Edinburg, TX 78539, USA
binfu@cs.panam.edu

Abstract

We investigate the complexity of integration and derivative for multivariate polynomials in the standard computation model. The integration is in the unit cube $[0, 1]^d$ for a multivariate polynomial, which has format $f(x_1, \dots, x_d) = p_1(x_1, \dots, x_d)p_2(x_1, \dots, x_d) \cdots p_k(x_1, \dots, x_d)$, where each $p_i(x_1, \dots, x_d) = \sum_{j=1}^d q_j(x_j)$ with all single variable polynomials $q_j(x_j)$ of degree at most two and constant coefficients. We show that there is no any factor polynomial time approximation for the integration $\int_{[0,1]^d} f(x_1, \dots, x_d) dx_1 \cdots dx_d$ unless $P = NP$. For the complexity of multivariate derivative, we consider the functions with the format $f(x_1, \dots, x_d) = p_1(x_1, \dots, x_d)p_2(x_1, \dots, x_d) \cdots p_k(x_1, \dots, x_d)$, where each $p_i(x_1, \dots, x_d)$ is of degree at most 2 and 0, 1 coefficients. We also show that unless $P = NP$, there is no any factor polynomial time approximation to its derivative $\frac{\partial f^{(d)}(x_1, \dots, x_d)}{\partial x_1 \cdots \partial x_d}$ at the origin point $(x_1, \dots, x_d) = (0, \dots, 0)$. Our $\#P$ -hard result for derivative shows that the derivative is not be easier than the integration in high dimension. We also give some tractable cases of high dimension integration and derivative.

1 Introduction

Integration and derivative are basic operations in the classical mathematics. Integrations with a large number of variables have been found applications in many areas such as finance, nuclear physics, and quantum system, etc. The complexity for approximating multivariate integration has been studied by measuring the number of function evaluations. For example, Sloan and Wozniakowski proved an exponential lower bounds 2^s of function evaluations in order to obtain an approximation with error less than the integration itself, which has s variables [9]. The integration $\int_{[0,1]^s} f(x_1, \dots, x_s) dx_1 \cdots dx_s$ is over the cubic $[0, 1]^s$ for some function $f(x_1, \dots, x_s)$. In the quasi-Monte Carlo method for computing $\int_{[0,1]^d} f(x) dx$, it is approximated by $\frac{1}{n} \sum_{i=1}^n f(x_i)$. This approximation has an error $\Theta(\frac{(\ln n)^{d-1}}{n})$, which grows exponentially on the dimension number d (see e.x., [7, 6] and the reference papers there).

An integration may be computed by the structure of the function without involving function evaluation. For example, $\int_{[0,1]^2} x^2 y^3 dx dy = (\int_{[0,1]} x^2 dx) \cdot (\int_{[0,1]} y^3 dy) = \frac{1}{3} \cdot \frac{1}{4} = \frac{1}{12}$. The calculation gives the exact value of the integration, but there is no evaluation for the function $f(x, y) = x^2 y^3$. Using the computational complexity theory, we study the polynomial time approximation limitation for the high dimensional integration for some easily defined functions. In this paper, we consider the high dimensional integration for multivariate polynomials, which are defined with format $f(x_1, \dots, x_d) =$

*This research is supported in part by National Science Foundation Early Career Award 0845376.

$p_1(x_1, \dots, x_d)p_2(x_1, \dots, x_d) \cdots p_k(x_1, \dots, x_d)$, where each $p_i(x_1, \dots, x_d) = \sum_{j=1}^d q_j(x_j)$ with polynomial $q_j(x_j)$ of constant degree. Its integration can be computed in polynomial space. We show how this problem is related to other hard problem in the field of computational complexity theory. Therefore, our model for studying the complexity of high dimensional integration is totally different from the existing approaches such as [9], and is more general than the old models. We show that there is no any factor polynomial time approximation to the integration problem unless $P = NP$.

A similar hardness of approximation result is also derived for the derivative of the polynomial function. The recent development of monomial testing theory [2, 3, 1] can be used to explain the hardness for computing the derivative for a $\prod \sum \prod$ polynomial. For the hardness of the approximation for multivariate derivative, we consider the functions with the format $f(x_1, \dots, x_d) = p_1(x_1, \dots, x_d)p_2(x_1, \dots, x_d) \cdots p_k(x_1, \dots, x_d)$, where each $p_i(x_1, \dots, x_d)$ is of degree 2. We also show that unless $P = NP$, there is no any factor polynomial time approximation to its derivative $\frac{\partial f^{(d)}(x_1, \dots, x_d)}{\partial x_1 \cdots \partial x_d}$ at the origin point $(x_1, \dots, x_d) = (0, \dots, 0)$. Our results show that the high dimension derivative may not be easier than the high dimension integration. Since both integration and derivative are widely used, this approach may help understand the complexity of some mathematics systems that involve high dimension integration or derivative.

Partial derivatives were used in developing deterministic algorithms for the polynomial identity problem (for example, see [8]), a fundamental problem in the computational complexity theory. Our intractability result for the high dimension derivative over multivariate polynomial points out a barrier of this approach.

Second part of this paper about the inapproximability of derivative is an application of our recently developed monomial testing theory [2, 3, 1]. It shows that it is $\#P$ -hard to compute the derivative of a $\prod \sum$ polynomial at the origin point. We also give some tractable cases of high dimension integration and derivative.

In section 3, we give an overview about our method for deriving the inapproximation result of high dimension integration. The main result of this paper is the inapproximation for high dimension integration, and is presented in section 4. In section 5, we present the inapproximation result for high order derivative. Some tractable cases of high dimension integration and derivative are shown in section 6.

2 Notations

Let $N = \{0, 1, 2, \dots\}$ be the set of all natural numbers. Let $N^+ = \{1, 2, \dots\}$ be the set of all positive natural numbers.

Assume that function $r(n)$ is from N to N^+ . For a functor $F(\cdot)$, which converts a multivariate polynomial into a real number, an algorithm $A(\cdot)$ gives an $r(n)$ -factor approximation to $F(f)$ if it satisfies the following conditions: if $F(f) \geq 0$, then $\frac{F(f)}{r(n)} \leq A(f) \leq r(n)F(f)$; and if $F(f) < 0$, then $r(n)F(f) \leq A(f) \leq \frac{F(f)}{r(n)}$, where n is the number of variables in f .

Assume that functions $r(n)$ and $s(n)$ are from N to N^+ . For a functor $F(\cdot)$, an algorithm $A(\cdot)$ gives an $(r(n), s(n))$ -factor approximation to $F(f)$ such that if $F(f) \geq 0$, then $\frac{F(f)}{r(n)} - s(n) \leq A(f) \leq r(n)F(f) + s(n)$; and if $F(f) < 0$, then $r(n)F(f) - s(n) \leq A(f) \leq \frac{F(f)}{r(n)} + s(n)$, where n is the number of variables in f .

In this paper, we consider two kinds of functors. The first one is the integration in the unit cube for a multivariate polynomial: $\int_{[0,1]^d} f(x_1, \dots, x_d) dx_1 \cdots dx_d$. The second is the derivative $\frac{\partial f^{(d)}(x_1, \dots, x_d)}{\partial x_1 \cdots \partial x_d}$ at the origin point $(x_1, \dots, x_d) = (0, \dots, 0)$.

For the complexity of multivariate integration, we consider the functions with the format below:

$$f(x_1, \dots, x_d) = p_1(x_1, \dots, x_d)p_2(x_1, \dots, x_d) \cdots p_k(x_1, \dots, x_d),$$

where each $p_i(x_1, \dots, x_d) = \sum_{j=1}^d q_j(x_j)$ with each single variable polynomials $q_j(x_j)$ of constant degree. This kind multivariate polynomial is called $\prod \sum S_c$ if the degree of each $q_j(x_j)$ is at most c .

For the complexity of multivariate derivative, we consider the functions with the format below:

$$f(x_1, \dots, x_d) = p_1(x_1, \dots, x_d)p_2(x_1, \dots, x_d) \cdots p_k(x_1, \dots, x_d),$$

where each $p_i(x_1, \dots, x_d)$ is of a constant degree. The polynomial $f(x_1, \dots, x_d)$ is called a $\prod \sum \prod_k$ polynomial if the degree of each $p_i(x_1, \dots, x_d)$ is at most k .

An algorithm is *subexponential time* if it runs in $2^{n^{o(1)}}$ time for all inputs of length n . Define subE to be the class of languages that have subexponential time algorithms.

3 Overview of Our Methods

In this section, we show the brief idea to derive the main result of this paper (Theorem 4). 3SAT is an NP-complete problem proved by Cook [4]. We show that approximating the integration of a $\prod \sum S_2$ polynomial is NP-hard by a reduction from 3SAT problem to it. It is still NP-hard to decide a conjunctive normal form that each variable appears at most three times with at most one negative time. We assume that each variable has its negation appears at most one time (Otherwise, we replace it by its negation).

We show (see Lemma 2 and equations (59) to (65) at its proof) that there exist integer coefficients polynomial functions $g_1(x) = ax^3 + bx^2 + cx + d$, $g_2(x) = ux + v$, and $f(x) = 2x$ satisfy that $\int_0^1 g_1(x)dx = 1$, $\int_0^1 g_2(x)dx = 1$, $\int_0^1 f(x)dx = 1$, $\int_0^1 g_1(x)g_2(x)dx = 4$, $\int_0^1 g_1(x)f(x)dx = 0$, $\int_0^1 g_2(x)f(x)dx = 0$, and $\int_0^1 g_1(x)g_2(x)f(x)dx = 0$.

Example 1. Consider the logical formula $F = (x_1 + x_2)(x_1 + \bar{x}_2)(\bar{x}_1 + x_2)$, which has the sum of product expansion $x_1x_1\bar{x}_1 + x_1x_1x_2 + x_1\bar{x}_2\bar{x}_1 + x_1\bar{x}_2x_2 + x_2x_1\bar{x}_1 + x_2x_1x_2 + x_2\bar{x}_2\bar{x}_1 + x_2\bar{x}_2x_2$. The term $x_1x_1x_2$ can bring a truth assignment $x_1 = \text{true}$ and $x_2 = \text{true}$ to make F true. As each variable appears at most 3 times with at most one negative appearance, the first positive x_i is replaced by $g_1(y_i)$, the second positive x_i is replaced by $g_2(y_i)$, and the negative \bar{x}_i is replaced by $f(y_i)$. It is converted into the polynomial

$$p(y_1, y_2) = (g_1(y_1) + g_1(y_2))(g_2(y_1) + f(y_2))(f(y_1) + g_2(y_2)).$$

The polynomial $p(y_1, y_2)$ has the sum of product expansion

$$\begin{aligned} &g_1(y_1)g_2(y_1)f(y_1) + g_1(y_1)g_2(y_1)g_2(y_2) + g_1(y_1)f(y_2)f(y_1) + g_1(y_1)f(y_2)g_2(y_2) + \\ &g_1(y_2)g_2(y_1)f(y_1) + g_1(y_2)g_2(y_1)g_2(y_2) + g_1(y_2)f(y_2)f(y_1) + g_1(y_2)f(y_2)g_2(y_2). \end{aligned}$$

Consider the integration $\int_{[0,1]^2} p(y_1, y_2) d_{y_1} d_{y_2}$. The integration can be distributed into those product terms. $\int_{[0,1]^2} g_1(y_1)g_2(y_1)g_2(y_2) d_{y_1} d_{y_2}$ is one of them. We have

$$\int_{[0,1]^2} g_1(y_1)g_2(y_1)g_2(y_2) d_{y_1} d_{y_2} = \left(\int_{[0,1]} g_1(y_1)g_2(y_1) d_{y_1} \right) \left(\int_{[0,1]} g_2(y_2) d_{y_2} \right) = 4 \cdot 1 = 4.$$

The integrations for other terms are all non-negative integers. Thus, $\int_{[0,1]^2} p(y_1, y_2) d_{y_1} d_{y_2}$ is a positive integer due to the satisfiability of F .

Example 2. Consider the logical formula $G = (x_1 + x_2)\bar{x}_1\bar{x}_2$, which has the sum of product expansion $x_1\bar{x}_1\bar{x}_2 + x_1\bar{x}_2\bar{x}_2$. Neither $x_1\bar{x}_1\bar{x}_2$ nor $x_1\bar{x}_2\bar{x}_2$ can be satisfied. As each variable appears at most 3 times with at most one negative appearance, the first positive x_i is replaced by $g_1(y_i)$, the second positive x_i is replaced by $g_2(y_i)$, and the negation case \bar{x}_i is replaced by $f(y_i)$. It is converted into the polynomial $q(y_1, y_2) = (g_1(y_1) + g_1(y_2))f(y_1)f(y_2)$. The polynomial $q(y_1, y_2)$ has the sum of product expansion $g_1(y_1)f(y_1)f(y_2) + g_1(y_2)f(y_1)f(y_2)$.

Consider the integration $\int_{[0,1]^2} q(y_1, y_2) d_{y_1} d_{y_2}$, which is identical to $\int_{[0,1]^2} g_1(y_1)f(y_1)f(y_2) d_{y_1} d_{y_2} + \int_{[0,1]^2} g_1(y_2)f(y_1)f(y_2) d_{y_1} d_{y_2}$. We have

$$\int_{[0,1]^2} g_1(y_1)f(y_1)f(y_2) d_{y_1} d_{y_2} = \left(\int_{[0,1]} g_1(y_1)f(y_1) d_{y_1} \right) \left(\int_{[0,1]} f(y_2) d_{y_2} \right) = 0 \cdot 1 = 0.$$

We also have

$$\int_{[0,1]^2} g_1(y_2)f(y_1)f(y_2)d_{y_1}d_{y_2} = \left(\int_{[0,1]} f(y_1)d_{y_1}\right)\left(\int_{[0,1]} g_1(y_2)f(y_2)d_{y_2}\right) = 1 \cdot 0 = 0.$$

Therefore, $\int_{[0,1]^2} q(y_1, y_2)d_{y_1}d_{y_2} = 0$ due to the unsatisfiability of G . Therefore, for any factor $a(n) > 0$, a polynomial time factor $a(n)$ -approximation to the integration of a $\prod \sum S_2$ polynomial implies a polynomial time decision for the satisfiability of the corresponding boolean formula.

4 Intractability of High Dimensional Integration

In this section, we show that the integration in high dimensional cube $[0, 1]^d$ does not have any factor approximation. We will reduce an existing NP-complete problem to the integration problem. Our main technical contribution is in converting a logical formula into a polynomial. We often use a basic property of integration, which can be found in some standard text books of calculus (for example [11]). Assume function $f(x_1, \dots, x_d) = f_1(x_{i_1}, \dots, x_{i_{d_1}})f_2(x_{j_1}, \dots, x_{j_{d_2}})$, where $\{x_1, \dots, x_d\}$ is the disjoint union of $\{x_{i_1}, \dots, x_{i_{d_1}}\}$ and $\{x_{j_1}, \dots, x_{j_{d_2}}\}$. Then we have

$$\int_{[0,1]^d} f(x_1, \dots, x_d)d_{x_1} \dots d_{x_d} \tag{1}$$

$$= \left(\int_{[0,1]^{d_1}} f_1(x_{i_1}, \dots, x_{i_{d_1}})d_{x_{i_1}} \dots d_{x_{i_{d_1}}}\right) \cdot \left(\int_{[0,1]^{d_2}} f_2(x_{j_1}, \dots, x_{j_{d_2}})d_{x_{j_1}} \dots d_{x_{j_{d_2}}}\right). \tag{2}$$

In order to make the conversion from logical operation to algebraic operation, we represent conjunctive normal form with the following format. For example, the formula $(x_1 + x_2)(x_1 + \bar{x}_2)(\bar{x}_1 + x_2)$ is a conjunctive normal form with two boolean variables x_1 and x_2 , where $+$ represents the logical \vee , and \cdot represent the logical \wedge .

Definition 1

- A 3SAT instance is a conjunctive form $C_1 \cdot C_2 \dots C_m$ such each C_i is a disjunction of at most three literals.
- 3SAT is the language of those 3SAT instances that have satisfiable assignments.
- A (3, 3)-SAT instance is an instance G for 3SAT such that for each variable x , the total number of times of x and \bar{x} in G is at most 3, and the total number of times of \bar{x} in G is at most 1.
- (3, 3)-SAT is the language of those (3, 3)-SAT instances that have satisfiable assignments.

For examples, $(x_1 + x_2 + x_3)(x_1 + \bar{x}_2)(\bar{x}_1 + x_2)$ is both 3SAT and (3, 3)-SAT instance, and also belongs to both 3SAT and (3, 3)-SAT. On the other hand, $(x_1 + x_2 + x_3)(\bar{x}_1 + \bar{x}_2)(\bar{x}_1 + x_2)$ is not a (3, 3)-SAT instance since \bar{x}_1 appears twice in the formula. The following lemma is similar to a result derived by Tovey [10].

Lemma 1 *There is a polynomial time reduction $f(\cdot)$ from 3SAT to (3, 3)-SAT.*

Proof Let F be an instance for 3SAT. Let's focus on one variable x_i that appears m times in F . Introduce a series of variables $y_{i,1}, \dots, y_{i,m}$ for x_i . Convert F to F' by changing the j -th occurrence of x_i in F to $y_{i,j}$ for $j = 1, \dots, m$. Define

$$\begin{aligned} G_{x_i} &= (x_i \rightarrow y_{i,1}) \cdot (y_{i,1} \rightarrow y_{i,2})(y_{i,2} \rightarrow y_{i,3}) \cdot (y_{i,3} \rightarrow y_{i,4}) \dots (y_{i,m-1} \rightarrow y_{i,m}) \cdot (y_{i,m} \rightarrow x_i) \\ &= (\bar{x}_i + y_{i,1}) \cdot (\bar{y}_{i,1} + y_{i,2}) \cdot (\bar{y}_{i,2} + y_{i,3}) \cdot (\bar{y}_{i,3} + y_{i,4}) \dots (\bar{y}_{i,m-1} + y_{i,m}) \cdot (\bar{y}_{i,m} + x_i). \end{aligned}$$

Each logical formula $(x \rightarrow y)$ is equivalent to $(\bar{x} + y)$. If G_{x_i} is true, then $x_i, y_{i,1}, \dots, y_{i,m}$ are equivalent.

Convert F' into F'' such that $F'' = F'G_{x_1} \cdots G_{x_k}$, where x_1, \dots, x_k are all variables in F .

For each variable x in F'' with more than one \bar{x} , create a new variable y_x , replace each positive x of F by \bar{y}_x , and each negative \bar{x} by y_x . Thus, F'' becomes F''' . It is easy to see that $F \in 3\text{SAT}$ iff F'' is satisfiable iff $F''' \in (3,3)\text{-SAT}$. \square

4.1 Integration of $\Pi \Sigma S_2$ Polynomial

Lemma 2 is our main technical lemma. It is used to convert a $(3,3)\text{-SAT}$ instance into a $\Pi \Sigma S_2$ polynomial.

Lemma 2 *There exist integers b, c, d, u , and v such that the functions $g_1(x) = bx^2 + cx + d$, $g_2(x) = ux + v$, and $f(x) = 2x$ satisfy that*

1. $\int_0^1 g_1(x)dx$, $\int_0^1 g_2(x)dx$, $\int_0^1 f(x)dx$, and $\int_0^1 g_1(x)g_2(x)dx$ are all positive integers, and
2. $\int_0^1 g_1(x)f(x)dx$, $\int_0^1 g_2(x)f(x)dx$, and $\int_0^1 g_1(x)g_2(x)f(x)dx$ are all equal to 0.

Proof We give the details how to derive the functions $g_1(x)$ and $g_2(x)$ to satisfy the conditions of the lemma. In order to avoid solving nonlinear equations, we will fix the two variables u and v in the early phase of the construction.

$$\int_{[0,1]} f(x)dx = \int_{[0,1]} 2xdx = x^2|_0^1 = 1. \quad (3)$$

$$\int_0^1 g_1(x)dx = \int_0^1 (bx^2 + cx + d)dx \quad (4)$$

$$= \left(\frac{bx^3}{3} + \frac{cx^2}{2} + dx \right) \Big|_0^1 \quad (5)$$

$$= \frac{b}{3} + \frac{c}{2} + d \quad (6)$$

$$= \frac{1}{6}(2b + 3c + 6d). \quad (7)$$

$$\int_0^1 g_2(x)dx = \int_0^1 (ux + v)dx \quad (8)$$

$$= \left(\frac{ux^2}{2} + vx \right) \Big|_0^1 \quad (9)$$

$$= \frac{1}{2}(u + 2v). \quad (10)$$

$$\int_0^1 g_2(x)f(x)dx = \int_0^1 (ux + v)2xdx \quad (11)$$

$$= 2 \int_0^1 (ux^2 + vx)dx \quad (12)$$

$$= 2 \left(\frac{ux^3}{3} + \frac{vx^2}{2} \right) \Big|_0^1 \quad (13)$$

$$= 2 \left(\frac{u}{3} + \frac{v}{2} \right) \quad (14)$$

$$= \frac{1}{3}(2u + 3v). \quad (15)$$

We let

$$u = -6 \quad (16)$$

$$v = 4. \quad (17)$$

Therefore, we have got

$$\int_{[0,1]} g_2(x) d_x = 1 \quad (\text{by equations (8) to (10)}), \text{ and} \quad (18)$$

$$\int_{[0,1]} g_2(x) f(x) d_x = 0 \quad (\text{by equations (11) to (15)}). \quad (19)$$

$$\int_0^1 g_1(x) g_2(x) d_x = \int_0^1 (bx^2 + cx + d)(ux + v) d_x \quad (20)$$

$$= \int_0^1 (bx^2 + cx + d)(-6x + 4) d_x \quad (21)$$

$$= \int_0^1 (-6bx^3 - 6cx^2 - 6dx + 4bx^2 + 4cx + 4d) d_x \quad (22)$$

$$= \int_0^1 ((-6b)x^3 + (4b - 6c)x^2 + (4c - 6d)x + 4d) d_x \quad (23)$$

$$= \left(\frac{(-6b)x^4}{4} + \frac{(4b - 6c)x^3}{3} + \frac{(4c - 6d)x^2}{2} + 4dx \right) \Big|_0^1 \quad (24)$$

$$= \left(\frac{(-6b)}{4} + \frac{(4b - 6c)}{3} + \frac{(4c - 6d)}{2} + 4d \right) \quad (25)$$

$$= \frac{1}{12} (3 \cdot (-6b) + 4 \cdot (4b - 6c) + 6 \cdot (4c - 6d) + 48d) \quad (26)$$

$$= \frac{1}{12} ((-18 + 16)b + (-24 + 24)c + (48 - 36)d) \quad (27)$$

$$= \frac{1}{12} ((-2)b + 12d) \quad (28)$$

$$= \frac{1}{6} (-b + 6d). \quad (29)$$

$$\int_0^1 g_1(x) f(x) d_x = \int_0^1 (bx^2 + cx + d) 2x d_x \quad (30)$$

$$= 2 \int_0^1 (bx^3 + cx^2 + dx) d_x \quad (31)$$

$$= 2 \left(\frac{bx^4}{4} + \frac{cx^3}{3} + \frac{dx^2}{2} \right) \Big|_0^1 \quad (32)$$

$$= 2 \left(\frac{b}{4} + \frac{c}{3} + \frac{d}{2} \right) \quad (33)$$

$$= \frac{1}{6} (3b + 4c + 6d). \quad (34)$$

$$\int_0^1 g_1(x) g_2(x) f(x) d_x \quad (35)$$

$$= \int_0^1 (bx^2 + cx + d)(ux + v)2xd_x \quad (36)$$

$$= 2 \int_0^1 (bx^2 + cx + d)(-6x + 4)xd_x \quad (37)$$

$$= 2 \int_0^1 (-6bx^3 - 6cx^2 - 6dx + 4bx^2 + 4cx + 4d)xd_x \quad (38)$$

$$= 2 \int_0^1 ((-6b)x^3 + (4b - 6c)x^2 + (4c - 6d)x + 4d)xd_x \quad (39)$$

$$= 2 \left(\frac{(-6b)x^5}{5} + \frac{(4b - 6c)x^4}{4} + \frac{(4c - 6d)x^3}{3} + \frac{4dx^2}{2} \right) \Big|_0^1 \quad (40)$$

$$= 2 \left(\frac{(-6b)}{5} + \frac{(4b - 6c)}{4} + \frac{(4c - 6d)}{3} + \frac{4d}{2} \right) \quad (41)$$

$$= \frac{2}{60} (12 \cdot (-6b) + 15 \cdot (4b - 6c) + 20 \cdot (4c - 6d) + 120d) \quad (42)$$

$$= \frac{1}{30} ((-72 + 60)b + (-90 + 80)c + (-120 + 120)d) \quad (43)$$

$$= \frac{1}{30} (-12b - 10c) \quad (44)$$

$$= \frac{1}{15} (-6b - 5c) \quad (45)$$

We need to satisfy the following conditions:

$$6b + 5c = 0 \quad (46)$$

$$3b + 4c + 6d = 0 \quad (47)$$

$$-b + 6d = 6n_1 \quad \text{for some positive integer } n_1 \quad (48)$$

$$2b + 3c + 6d = 6n_2 \quad \text{for some positive integer } n_2 \quad (49)$$

Equation (46) makes $\int_{[0,1]} g_1(x)g_2(x)f(x)d_x = 0$ according to equations (35) to (45). Equation (47) makes $\int_{[0,1]} g_1(x)f(x)d_x = 0$ according to equations (30) to (34). Equation (48) makes $\int_{[0,1]} g_1(x)g_2(x)d_x$ be a positive integer according to equations (20) to (29). Equation (49) makes $\int_{[0,1]} g_1(x)d_x$ be a positive integer according to equations (4) to (7).

Let x and k be integer parameters to be fixed later. We have the solutions below:

$$b = 5x \cdot 6^k, \quad (50)$$

$$c = -\frac{6b}{5} = -6x \cdot 6^k, \quad (\text{by equation (46)}) \quad \text{and} \quad (51)$$

$$d = \frac{1}{6}(-3b - 4c) = \frac{1}{6}(-3 \cdot (5x \cdot 6^k) - 4(-6x \cdot 6^k)) \quad (52)$$

$$= 9x \cdot 6^{k-1} \quad (\text{by equation (47)}). \quad (53)$$

We have the equations:

$$-b + 6d = -5x \cdot 6^k + 6 \cdot (9x \cdot 6^{k-1}) = 4x \cdot 6^k \quad \text{and} \quad (54)$$

$$2b + 3c + 6d = 2 \cdot (5x \cdot 6^k) + 3 \cdot (-6x \cdot 6^k) + 6 \cdot (9x \cdot 6^{k-1}) \quad (55)$$

$$= (10x - 18x + 9x)6^k = x \cdot 6^k. \quad (56)$$

Let $x = 1$ and $k = 1$. We have $b = 30$, $c = -36$, and $d = 9$. We also have

$$-b + 6d = 24 \quad (\text{by equation (54)}) \quad (57)$$

$$2b + 3c + 6d = 6 \quad (\text{by equation (56)}). \quad (58)$$

Thus, $g_1(x) = bx^2 + cx + d = 30x^2 - 36x + 9$, $g_2(x) = -6x + 4$, and $f(x) = 2x$. We have the following equations to satisfy the conditions in the lemma.

$$\int_0^1 f(x) d_x = 1, \quad (\text{by equation (3)}) \quad (59)$$

$$\int_0^1 g_1(x) d_x = 1, \quad (\text{by equations (4) to (7), and (58)}) \quad (60)$$

$$\int_0^1 g_2(x) d_x = 1, \quad (\text{by equation (18)}) \quad (61)$$

$$\int_0^1 g_1(x) g_2(x) d_x = 4, \quad (\text{by equations (20) to (29), and (57)}) \quad (62)$$

$$\int_0^1 g_1(x) f(x) d_x = 0, \quad (\text{by equations (30) to (34), and (47)}) \quad (63)$$

$$\int_0^1 g_2(x) f(x) d_x = 0, \quad (\text{by equation (19)}), \quad \text{and} \quad (64)$$

$$\int_0^1 g_1(x) g_2(x) f(x) d_x = 0. \quad (\text{by equations (35) to (45), and the solutions for } b \text{ and } c)) \quad (65)$$

□

Lemma 3 *There is a polynomial time algorithm h such that given a $(3, 3)$ -SAT instance $s(x_1, \dots, x_n)$, it produces a $\prod \sum S_2$ polynomial $h(s(x_1, \dots, x_n)) = p(y_1, \dots, y_n)$ to satisfy the following two conditions:*

1. *if $s(x_1, \dots, x_n)$ is satisfiable, then $\int_{[0,1]^n} p(y_1, \dots, y_n) d_{y_1} \dots d_{y_n}$ is a positive integer; and*
2. *if $s(x_1, \dots, x_n)$ is not satisfiable, then $\int_{[0,1]^n} p(y_1, \dots, y_n) d_{y_1} \dots d_{y_n}$ is zero.*

Proof We give two examples to show how a logical formula is converted into a multivariate polynomial in section 3. Let polynomials $g_1(y)$, $g_2(y)$, and $f(y)$ be defined according to those in Lemma 2.

For a $(3, 3)$ -SAT problem $s(x_1, \dots, x_n)$, let $p(y_1, \dots, y_n)$ be defined as follows.

- For the first positive literal x_i in $s(x_1, \dots, x_n)$, replace it with $g_1(y_i)$.
- For the second positive literal x_i in $s(x_1, \dots, x_n)$, replace it with $g_2(y_i)$.
- For the negative literal \bar{x}_i in $s(x_1, \dots, x_n)$, replace it with $f(y_i)$.

The formula $s(x_1, \dots, x_n)$ has a sum of product form. It is satisfiable if and only if one term does not contain a positive and negative literals for the same variable. If a term contains both x_i and \bar{x}_i , the corresponding term in the sum of product for $p(\cdot)$ contains both $g_j(y_i)$ and $f(y_i)$ for some $j \in \{1, 2\}$. This makes it zero after integration by Lemma 2. Therefore, $s(x_1, \dots, x_n)$ is satisfiable if and only if $\int_{[0,1]^n} p(y_1, \dots, y_n) d_{y_1} \dots d_{y_n}$ is not zero. Furthermore, it is satisfiable, the integration is a positive integer by Lemma 2. See the two examples in section 3. The computational time of h is clearly polynomial since we convert s to $h(s)$ by replacing each literal by a single variable function of degree at most 2. □

Theorem 4 *Let $a(n)$ be an arbitrary function from N to N^+ . Then there is no polynomial time $a(n)$ -factor approximation for the integration of a $\prod \sum S_2$ polynomial $p(x_1, \dots, x_n)$ in the region $[0, 1]^n$ unless $P = NP$.*

Proof Assume that $A(\cdot)$ is a polynomial time $a(n)$ -factor approximation for the integration $\int_{[0,1]^n} p(y_1, \dots, y_n) d_{y_1} \dots d_{y_n}$ with $\prod \sum S_2$ polynomial $p(y_1, \dots, y_n)$. For a (3,3)-SAT instance $s(x_1, \dots, x_n)$, let $p(y_1, \dots, y_n) = h(s(x_1, \dots, x_n))$ according to Lemma 3. By Lemma 3, a (3,3)-SAT instance $s(x_1, \dots, x_n)$ is satisfiable if and only if the integration $J = \int_{[0,1]^n} p(y_1, \dots, y_n) d_{y_1} \dots d_{y_n}$ is not zero. Assume that $s(x_1, \dots, x_n)$ is not satisfiable, then we have $A(J) \in [J/a(n), J \cdot a(n)] = [0, 0]$, which implies $A(J) = 0$. Assume that $s(x_1, \dots, x_n)$ is satisfiable, then we have $A(J) \in [J/a(n), J \cdot a(n)] \subseteq (0, +\infty)$, which implies $A(J) > 0$. Thus, $s(x_1, \dots, x_n)$ is satisfiable if and only if $A(J) > 0$.

Therefore, there is a polynomial time algorithm for solving (3,3)-SAT, which is NP-complete by Lemma 1. So, $P = NP$. \square

Theorem 5 *Let $a(n)$ be an arbitrary function from N to N^+ . Then there is no subexponential time $a(n)$ -factor approximation for the integration of a $\prod \sum S_2$ polynomial $p(x_1, \dots, x_n)$ in the region $[0, 1]^n$ unless $NP \subseteq \text{subE}$.*

Proof Assume that $A(\cdot)$ is a subexponential time $a(n)$ -factor approximation for the integration $\int_{[0,1]^n} p(y_1, \dots, y_n) d_{y_1} \dots d_{y_n}$ with $\prod \sum S_2$ polynomial $p(y_1, \dots, y_n)$.

For a (3,3)-SAT instance $s(x_1, \dots, x_n)$, let $p(y_1, \dots, y_n) = h(s(x_1, \dots, x_n))$ according to Lemma 3. By Lemma 3, a (3,3)-SAT instance $s(x_1, \dots, x_n)$ is satisfiable if and only if the integration $J = \int_{[0,1]^n} p(y_1, \dots, y_n) d_{y_1} \dots d_{y_n}$ is not zero. Assume that $s(x_1, \dots, x_n)$ is not satisfiable, then we have $A(J) \in [J/a(n), J \cdot a(n)] = [0, 0]$, which implies $A(J) = 0$. Assume that $s(x_1, \dots, x_n)$ is satisfiable, then we have $A(J) \in [J/a(n), J \cdot a(n)] \subseteq (0, +\infty)$, which implies $A(J) > 0$. Thus, $s(x_1, \dots, x_n)$ is satisfiable if and only if $A(J) > 0$.

Therefore, there is a subexponential time algorithm for solving (3,3)-SAT, which is NP-complete by Lemma 1. Thus, $NP \subseteq \text{subE}$. \square

Lemma 6 *Assume that $a(1^n)$ is a polynomial time computable function from N to N^+ with $a(1^n) > 0$ for n . There is a polynomial time algorithm such that given a (3,3)-SAT instance $s(x_1, \dots, x_n)$, it generates a $\prod \sum S_2$ polynomial $p(y_1, \dots, y_n)$ such that if $s(x_1, \dots, x_n)$ is satisfiable, then $\int_{[0,1]^n} p(y_1, \dots, y_n) d_{y_1 \dots y_n}$ is a positive integer at least $3a(1^n)^2$; and if $s(x_1, \dots, x_n)$ is not satisfiable, $\int_{[0,1]^n} p(y_1, \dots, y_n) d_{y_1 \dots y_n}$ is zero.*

Proof For a (3,3)-SAT problem $s(x_1, \dots, x_n)$, let $q(y_1, \dots, y_n) = h(s(x_1, \dots, x_n))$ be constructed as Lemma 3.

Since $a(1^n)$ is polynomial time computable, let $p(y_1, \dots, y_n) = 3a(1^n)^2 q(y_1, \dots, y_n)$, which can be computed in a polynomial time. \square

Theorem 7 *Let $a(1^n)$ be a polynomial time computable function from N to N^+ . Then there is no polynomial time $(a(1^n), a(1^n))$ -approximation for the integration problem $\int_{[0,1]} f(x_1, \dots, x_d) d_{x_1} \dots d_{x_d}$ for a $\prod \sum S_2$ polynomial $f(\cdot)$ unless $P = NP$.*

Proof Assume that there is a polynomial time $(a(1^n), a(1^n))$ -approximation $App(\cdot)$ for the integration problem $\int_{[0,1]} f(x_1, \dots, x_d) d_{x_1} \dots d_{x_d}$ for a $\prod \sum S_2$ polynomial $f(\cdot)$.

Let $s(x_1, \dots, x_n)$ be an arbitrary (3,3)-SAT instance. Let $p(y_1, \dots, y_n)$ be the polynomial according to Lemma 6.

Let $J = \int_{[0,1]^n} p(y_1, \dots, y_n) d_{y_1 \dots y_n}$. If $s(x_1, \dots, x_n)$ is not satisfiable, then $J = 0$. Otherwise, $J \geq 3a(1^n)^2$.

Assume that $s(x_1, \dots, x_n)$ is not satisfiable. Since $App(J)$ is an $(a(1^n), a(1^n))$ -approximation, we have $App(J) \leq J \cdot a(1^n) + a(1^n) = a(1^n)$ by the definition in section 2.

Assume that $s(x_1, \dots, x_n)$ is satisfiable. Since $App(J)$ is an $(a(1^n), a(1^n))$ -approximation, we have $App(J) \geq \frac{J}{a(1^n)} - a(1^n) \geq \frac{3a(1^n)^2}{a(1^n)} - a(1^n) = 2a(1^n)$ by the definition in section 2.

Therefore, $s(x_1, \dots, x_n)$ is satisfiable if and only if $App(J) \geq 2a(1^n)$. Thus, if there is a polynomial time $(a(1^n), a(1^n))$ -approximation, then there is a polynomial time algorithm for solving (3,3)-SAT. By Lemma 1, $P = NP$. \square

The well known exponential time hypothesis says $NP \not\subseteq subE$ [5]. Basing on such a hypothesis, we have the following stronger result about the intractability of high dimension integration.

Theorem 8 *Let $a(1^n)$ be a polynomial time computable function from N to N^+ . Then there is no subexponential time $(a(1^n), a(1^n))$ -approximation for the integration problem $\int_{[0,1]} f(x_1, \dots, x_d) dx_1 \dots dx_d$ with a $\prod \sum S_2$ polynomial $f(\cdot)$ unless $NP \subseteq subE$.*

Proof Assume that there is a subexponential time $(a(1^n), a(1^n))$ -approximation $App(\cdot)$ for the integration problem $\int_{[0,1]} f(x_1, \dots, x_d) dx_1 \dots dx_d$ with a $\prod \sum S_2$ polynomial $f(\cdot)$.

Let $s(x_1, \dots, x_n)$ be an arbitrary (3,3)-SAT instance. Let $p(y_1, \dots, y_n)$ be the polynomial according to Lemma 6. Let $J = \int_{[0,1]^n} p(y_1, \dots, y_n) dy_1 \dots dy_n$. By Lemma 6, we have $J \geq 0$.

If $s(x_1, \dots, x_n)$ is not satisfiable, then $J = 0$. Otherwise, $J \geq 3a(1^n)^2$.

Assume that $s(x_1, \dots, x_n)$ is not satisfiable. Since $App(J)$ is an $(a(1^n), a(1^n))$ -approximation, we have $App(J) \leq J \cdot a(1^n) + a(1^n) = a(1^n)$ by the definition in section 2.

Assume that $s(x_1, \dots, x_n)$ is satisfiable. Since $App(J)$ is an $(a(1^n), a(1^n))$ -approximation, we have $App(J) \geq \frac{J}{a(1^n)} - a(1^n) \geq \frac{3a(1^n)^2}{a(1^n)} - a(1^n) = 2a(1^n)$ by the definition in section 2.

Therefore, $s(x_1, \dots, x_n)$ is satisfiable if and only if $App(J) \geq 2a(1^n)$. Thus, if there is a subexponential time $(a(1^n), a(1^n))$ -approximation, then there is a subexponential time algorithm for solving (3,3)-SAT. By Lemma 1, $NP \subseteq subE$. \square

5 Inapproximation of Derivative

In this section, we study the hardness of high dimensional derivative. We derive the inapproximation results under both $NP \neq P$ and $NP \not\subseteq subE$ assumptions.

Definition 2 *A monomial is an expression $x_1^{a_1} \dots x_d^{a_d}$ and its degree is $a_1 + \dots + a_d$. A monomial $x_1^{a_1} \dots x_d^{a_d}$, in which x_1, \dots, x_d are different variables, is a multilinear if $a_1 = a_2 = \dots = a_d = 1$.*

For example, $(x_1 x_3 + x_2^2)(x_2 x_4 + x_3^2)$ is a $\prod \sum \prod_2$ polynomial. It has a multilinear monomial $x_1 x_2 x_3 x_4$ in its sum of products expansion.

We give Lemma 9 to convert an instance f for (3,3)-SAT into a $\prod \sum \prod_2$ polynomial. The technology developed in [2, 1] will be applied in the construction.

Lemma 9 *Let $a(1^n)$ be a polynomial time computable function from N to N^+ . Then there is a polynomial time algorithm A such that given a (3,3)-SAT instance $F(y_1, \dots, y_d)$, the algorithm returns a $\prod \sum \prod_2$ polynomial $G(x_1, \dots, x_n)$ such that*

1. *If F is not satisfiable, then G does not have a multilinear monomial with a nonzero coefficient in its sum of product expansion.*
2. *If F is satisfiable, then G has the multilinear monomial $x_1 \dots x_n$ with a positive integer coefficient at least $3a(1^n)^2$ in its sum of product expansion.*

Proof Let $(3,3)$ -SAT instance F be $C_1 C_2 \cdots C_k$. Each clause C_i has format $y_{i_1}^* + y_{i_2}^* + y_{i_3}^*$, where literal y_j^* is either y_j or its negation \bar{y}_j . Since F is a $(3,3)$ -SAT instance, for each variable y_i in F , y_i and \bar{y}_i totally appear at most three times in F , and \bar{y}_i appears at most once in F .

For each variable y_i in F , create four new variables $z_{i,1}$, $z_{i,2}$, $u_{i,1}$ and $u_{i,2}$. Convert formula F into polynomial G_1 such that for each y_i in F , the first positive occurrence y_i is changed into $z_{i,1}u_{i,1}$, the second positive occurrence y_i is changed into $z_{i,2}u_{i,2}$, and the negative occurrence \bar{y}_i is changed to $z_{i,1}z_{i,2}$. After the conversion for all the variables, formula F is transformed into a polynomial G_1 . We have that F is satisfiable if and only if G_1 has a multilinear monomial with positive coefficient in its sum of products expansion. This is because a multilinear monomial in the sum of product expansion of G_1 corresponds a consistent conjunctive term, which does not contain both y_i and its negation \bar{y}_i for some variable y_i , in the sum of product expansion of F .

Let H_1 be the set of all variables in G_1 . Assume that d_1 is the degree of G_1 (it is easy to see that all monomials in the sum-product expansion of G_1 have the same degree d_1). Let m be the number of variables in H_1 . Let d be the number of boolean variables in F . Assume that no clause C_i in F contains a single literal (otherwise, we can force the literal to be true to simplify F). The number of clauses in F is at most $\frac{3d}{2}$ since each variable appears in F at most three times and each clause C_i of F contains at least two literals. The degree G_1 is at most $3d$ since each literal of F is replaced by a product of two variables. The number of variables in G_1 is $m = 4d$ (we create four new variables for each variable in F), which is larger than the degree of G_1 .

Create new variables v_1, \dots, v_{m-d_1} . For $j = 1, \dots, m - d_1$, let $q_j = \sum_{x \in H_1} x v_j$. Finally, we get the polynomial $G = 3a(1^n)^2 \cdot G_1 \cdot q_1 \cdots q_{m-d_1}$, where $n = m + (m - d_1)$. Note that $3a(1^n)^2$ in the polynomial G is considered an integer constant which does not contain any variable. The degree of G is $n = d_1 + 2(m - d_1) = m + (m - d_1)$. Thus, the degree of G is the same as the total number of variables in g . We can show that f is satisfiable if and only if there is a multilinear monomial, which is the product of all variables in G , with positive coefficient of size at least $3a(1^n)^2$. \square

Theorem 10 Assume that $r(n)$ is a function from N to N^+ . If there is a polynomial time algorithm A such that given a $\prod \sum \prod_2$ polynomial $g(x_1, \dots, x_n)$, it gives an $r(n)$ -factor approximation to $\frac{\partial g^{(n)}(x_1, \dots, x_n)}{\partial x_1 \cdots \partial x_n}$ at the origin point $(x_1, \dots, x_n) = (0, \dots, 0)$, then $P = NP$.

Proof Assume that $A(\cdot)$ is a polynomial time $r(n)$ -approximation for computing $\frac{\partial g^{(n)}(x_1, \dots, x_n)}{\partial x_1 \cdots \partial x_n}$ at the origin point $(x_1, \dots, x_n) = (0, \dots, 0)$.

Assume that f is an arbitrary formula in a $(3,3)$ -SAT problem. By Lemma 9, we can get a polynomial $g(x_1, \dots, x_n)$. The derivative $\frac{\partial g^{(n)}(0, \dots, 0)}{\partial x_1 \cdots \partial x_n}$ is equal to the coefficient of $x_1 \cdots x_n$ in the sum of product expansion of g .

If f is satisfiable, we have $A(g) > 0$, and if f is not satisfiable, we have $A(g) = 0$ since $A(\cdot)$ is a $r(n)$ -approximation and $r(n) \geq 1$. We can know if the coefficient of $x_1 \cdots x_n$ in the sum of product expansion of g is positive in polynomial time. Thus, $(3,3)$ -SAT is solvable in polynomial time. Since $(3,3)$ -SAT is NP-complete, we have $P = NP$. \square

Basing on exponential time hypothesis $NP \not\subseteq \text{subE}$ [5], we have the following stronger result about the intractability of high dimension derivative.

Theorem 11 Assume that $r(n)$ is a function from N to N^+ . If there is a subexponential time algorithm A such that given a $\prod \sum \prod_2$ polynomial $g(x_1, \dots, x_n)$, it gives an $r(n)$ -factor approximation to $\frac{\partial g^{(n)}(x_1, \dots, x_n)}{\partial x_1 \cdots \partial x_n}$ at the origin point $(x_1, \dots, x_n) = (0, \dots, 0)$, then $NP \subseteq \text{subE}$.

Proof Assume that $A(\cdot)$ is a subexponential time $r(n)$ -approximation for computing $\frac{\partial g^{(n)}(x_1, \dots, x_n)}{\partial x_1 \cdots \partial x_n}$ at the origin point $(x_1, \dots, x_n) = (0, \dots, 0)$.

Assume that f is an arbitrary formula in a $(3, 3)$ -SAT problem. By Lemma 9, we can get a polynomial $g(x_1, \dots, x_n)$. The derivative $\frac{\partial g^{(n)}(0, \dots, 0)}{\partial x_1 \dots \partial x_n}$ is equal to the coefficient of $x_1 \dots x_n$ in the sum of product expansion of g .

If f is satisfiable, we have $A(g) > 0$, and if f is not satisfiable, we have $A(g) = 0$ since $A(\cdot)$ is a $r(n)$ -approximation and $r(n) \geq 1$. We can know if the coefficient of $x_1 \dots x_n$ in the sum of product expansion of g is positive in subexponential time. Thus, $(3, 3)$ -SAT is solvable in subexponential time. Since $(3, 3)$ -SAT is NP-complete, we have $\text{NP} \subseteq \text{subE}$. \square

Theorem 12 *Let $a(1^n)$ be a polynomial time computable function from N to N^+ . Then there is no polynomial time $(a(1^n), a(1^n))$ -approximation for $\frac{\partial g^{(n)}(x_1, \dots, x_n)}{\partial x_1 \dots \partial x_n}$ with $g(x_1, \dots, x_n)$ as a $\Pi \Sigma \Pi_2$ polynomial at the origin point $(x_1, \dots, x_n) = (0, \dots, 0)$, unless $\text{P} = \text{NP}$.*

Proof Assume that $\text{App}(\cdot)$ is a polynomial time $(a(1^n), a(1^n))$ -approximation for computing $\frac{\partial g^{(n)}(x_1, \dots, x_n)}{\partial x_1 \dots \partial x_n}$ at the origin point $(x_1, \dots, x_n) = (0, \dots, 0)$.

Given an arbitrary $(3, 3)$ -SAT instance F , let the $\Pi \Sigma \Pi_2$ polynomial $G(x_1, \dots, x_n)$ be constructed according to Lemma 9.

If F is not satisfiable, then we have $\frac{\partial G^{(n)}(x_1, \dots, x_n)}{\partial x_1 \dots \partial x_n}$ is zero at the origin point $(x_1, \dots, x_n) = (0, \dots, 0)$. Otherwise, it is at least $3a(1^n)^2$.

Assume that F is not satisfiable. Since $\text{App}(J)$ is an $(a(1^n), a(1^n))$ -approximation, we have $\text{App}(J) \leq J \cdot a(1^n) + a(1^n) = a(1^n)$ by the definition in section 2.

Assume that F is satisfiable. Since $\text{App}(J)$ is an $(a(1^n), a(1^n))$ -approximation, we have $\text{App}(J) \geq \frac{J}{a(1^n)} - a(1^n) \geq \frac{3a(1^n)^2}{a(1^n)} - a(1^n) = 2a(1^n)$ by the definition in section 2.

Therefore, F is satisfiable if and only if $\text{App}(J) \geq 2a(1^n)$. Thus, if there is a polynomial time $(a(1^n), a(1^n))$ -approximation, then there is a polynomial time algorithm for solving $(3, 3)$ -SAT. By Lemma 1, $\text{P} = \text{NP}$. \square

Theorem 13 *Let $a(1^n)$ be a polynomial time computable function from N to N^+ . Then there is no subexponential time $(a(1^n), a(1^n))$ -approximation for $\frac{\partial g^{(n)}(x_1, \dots, x_n)}{\partial x_1 \dots \partial x_n}$ at the origin point $(x_1, \dots, x_n) = (0, \dots, 0)$, unless $\text{NP} \subseteq \text{subE}$.*

Proof The proof is similar to that of Theorem 12. \square

6 Some Tractable Integrations and Derivatives

In this section, we present some polynomial time algorithms for integration with some restrictions. We also show a case that the derivative can fully polynomial time approximation scheme.

6.1 Bounded Width Product

Definition 3 *A formula $f_1 \cdot f_2 \dots f_m$ is c -wide if for each variables x_i , there is an index j such that x_i only appears in $f_j, f_{j+1}, \dots, f_{j+c-1}$, where each f_i is a sum of monomials.*

Theorem 14 *There is an $O(mn^{3c})$ time algorithm to compute the integration $\int_{[0,1]^d} F(x_1, \dots, x_d)$ for a c -wide formula $F(x_1, \dots, x_d) = f_1 \dots f_m$, where n is the total length of F .*

Proof Apply the divide and conquer method. Convert F into F_1GF_2 such that G is a product of at most c sub-formulas $f_i \cdots f_j$ with $j - i = c$ in the middle region of F (we can let $i = \lceil \frac{m-c}{2} \rceil + 1$, and $j = \lceil \frac{m-c}{2} \rceil + c$).

Let S_1 be the set of variables that are only in F_1 , S_2 be the set of variables that are only in F_2 , and S be the set of variables that appear in G . The set of variables in F is partitioned into S_1 , S , and S_2 .

As $F_1 = f_1 \cdots f_{i-1}$, we convert F_1 into $F_1^* = f_1 \cdots f_{i-c} f_1^*$, where f_1^* is the product of the last c sub-formulas: $f_1^* = f_{i-c} \cdots f_{i-1}$. Similarly, as $F_2 = f_{j+1} \cdots f_m$, we convert F_2 into $F_2^* = f_2^* f_{j+c} \cdots f_m$, where f_2^* is the product of the first c sub-formulas: $f_2^* = f_{j+1} \cdots f_{j+c}$. Convert G into the sum of products.

We have

$$\begin{aligned} \int_{[0,1]^d} F(x_1, \dots, x_d) d_{x_1} \cdots d_{x_d} &= \int_{[0,1]^{|S|}} G \cdot \left(\int_{[0,1]^{|S_1|}} F_1 d_{S_1} \right) \cdot \left(\int_{[0,1]^{|S_2|}} F_2 d_{S_2} \right) d_S \\ &= \int_{[0,1]^{|S|}} G \cdot \left(\int_{[0,1]^{|S_1|}} F_1^* d_{S_1} \right) \cdot \left(\int_{[0,1]^{|S_2|}} F_2^* d_{S_2} \right) d_S. \end{aligned}$$

The integration $\int_{[0,1]^{|S_1|}} F_1 d_{S_1}$ can be expressed as a polynomial of variables in S . The integration $\int_{[0,1]^{|S_2|}} F_2 d_{S_2}$ can be expressed as a polynomial of variables in S .

We have the recursive equation for the computational time $T(m) = 2T(m/2) + O(n^{3c})$. This gives $T(m) = O(mn^{3c})$. □

6.2 Tractable Derivative

In this section, we show that computing the derivative of a class of $\prod \sum$ polynomial is $\#P$ -hard, and also give a polynomial time randomized approximation scheme by using the theory of testing monomials developed by Chen and Fu [2, 1].

Definition 4 Let $f(x_1, \dots, x_d) = p_1(x_1, \dots, x_d) \cdots p_k(x_1, \dots, x_d)$ be a $\prod \sum$ polynomial. If for each $p_i(x_1, \dots, x_d)$, each variable's coefficient is either 0 or 1, then f is called a $\prod \sum^*$ polynomial.

We show that the derivative for a $\prod \sum^*$ polynomial has a polynomial time approximation scheme. Chen and Fu derived the following theorem by a reduction from the number of perfect matchings in a bipartite.

Theorem 15 (Chen and Fu [1])

1. There is a polynomial time randomized algorithm to approximate the coefficient of a $\prod \sum^*$ polynomial.
2. It is $\#P$ -hard to compute the coefficient of the multilinear $x_1 \cdots x_d$ in a $\prod \sum^*$ polynomial $f(x_1, \dots, x_d)$.

Theorem 15 implies Theorem 16.

Theorem 16

1. Let ϵ be an arbitrary constant in $(0, 1)$. Then there is a polynomial time randomized algorithm that given a $\prod \sum^*$ polynomial f , it returns a $(1 + \epsilon)$ -approximation for $\frac{\partial f(x_1, \dots, x_d)^{(d)}}{\partial x_1 \cdots \partial x_d}$ at the point $(0, \dots, 0)$.
2. It is $\#P$ -hard to compute $\frac{\partial f(x_1, \dots, x_d)^{(d)}}{\partial x_1 \cdots \partial x_d}$ at the point $(0, \dots, 0)$ for a $\prod \sum^*$ polynomial f .

Proof For a $\prod \sum^*$ polynomial $f(x_1, \dots, x_d)$, its $\frac{\partial f(x_1, \dots, x_d)^{(d)}}{\partial x_1 \cdots \partial x_d}$ at the point $(0, \dots, 0)$ is identical to the coefficient of the monomial $x_1 \cdots x_d$ in the sum of products in the expansion of $f(x_1, \dots, x_d)$. The theorem follows from Theorem 15. □

7 Conclusions

Using the theory of NP-hardness, we characterize the intractability of approximation for two fundamental mathematical operations: Integration and derivative in high dimensional space. We may see that this approach will be applied to determining the computational complexity of more mathematics systems that involve integration and derivative. We show that derivative for $\Pi \Sigma \Pi_2$ is $\#P$ -hard. Both integration and derivative for $\Pi \Sigma \Pi$ polynomials are in the class $\#P$. This shows that derivative is not easier than integration in the high dimension.

References

- [1] Z. Chen and B. Fu. Approximating multilinear monomial coefficients and maximum multilinear monomials in multivariate polynomials. Electronic Colloquium on Computational Complexity, ECCC-TR10-124, 2010.
- [2] Z. Chen and B. Fu. The complexity of testing monomials in multivariate polynomials. Electronic Colloquium on Computational Complexity, ECCC-TR10-114, 2010.
- [3] Z. Chen, B. Fu, Y. Liu, and R. Schweller. Algorithms for testing monomials in multivariate polynomials. Electronic Colloquium on Computational Complexity, ECCC-TR10-114, 2010.
- [4] S. A. Cook. The complexity of theorem-proving procedures. In *STOC*, pages 151–158, 1971.
- [5] R. Impagliazzo and R. Paturi. The complexity of k-sat. In *Proceedings of the 14th IEEE Conference on Computational Complexity*, page 237–240, 1999.
- [6] H. Niederreiter. Quasi-monte carlo methods and pseudo-random numbers. *Bulletin of the American Mathematical Society*, 84(6):957–1041, 1978.
- [7] H. Niederreiter. *Random Number Generation and quasi-Monte Carlo Methods*, volume 63. SIAM, Philadelphia, 1992.
- [8] A. Shpilka and I. Volkovich. Read-once polynomial identity testing. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pages 507–516, 2008.
- [9] I. H. Sloan and H. Wozniakowski. An intractability result for multiple integration. *Math. Comput.*, 66(219):1119–1124, 1997.
- [10] C. A. Tovey. A simplified satisfiability problem. *Discrete Applied Mathematics*, 8:85–89, 1984.
- [11] W. F. Trench. *Advanced Calculus*. Harper & Row, New York, 1978.